PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G06F 9/44 | A1

(11) International Publication Number: WO 94/11811

(43) International Publication Date: 26 May 1994 (26.05.94)

(21) International Application Number: PCT/US93/11061

(22) International Filing Date: 15 November 1993 (15.11.93)

(30) Priority data:
07/976,445 13 November 1992 (13.11.92) US

(71) Applicant: MICROSOFT CORPORATION [US/US]; One Microsoft Way, Redmond, WA 98052-6399 (US).

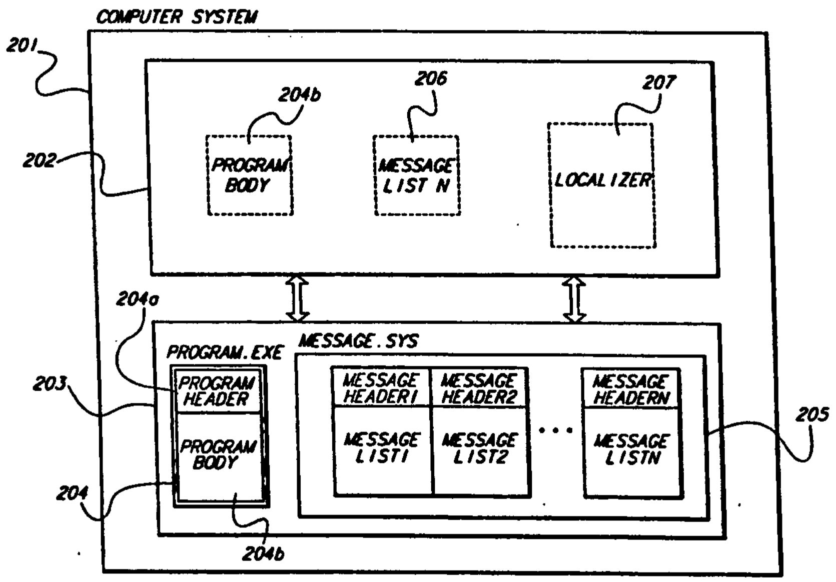
(72) Inventor: JAIN, Naveen, K. ; 2635 236th Place N.E., Redmond, WA 98053 (US).

(74) Agents: CHAMBERLAIN, Patricia, E. et al.; Seed and Berry, 6300 Columbia Center, 701 Fifth Avenue, Seattle, WA 98104-7092 (US).

(81) Designated States: CA, JP, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: MULTI-LINGUAL COMPUTER PROGRAMS



(57) Abstract

A method and system for creating multi-lingual computer programs by dynamically loading messages is provided. In a preferred embodiment, a user specifies a preferred language in which the computer program will communicate. The computer program has one or more associated message sets, each set in a unique natural language. At least some of the message sets are preferably stored in a message file on the computer system. After the computer program is invoked, a Localizer searches the message file for a message set in the preferred language. After selecting the message set, the Localizer makes a memory allocation request, loads the selected message set into the allocated memory, passes the address of the allocated memory to the computer program, and returns control to the computer program.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	GB	United Kingdom	MR	Mauritania
AU	Australia	GE	Georgia	MW	Malawi
BB	Barbados	GN	Guinea	NE	Niger
BE	Belgium	GR	Greece	NL	Netherlands
BF	Burkina Faso	HU	Hungary	NO	Norway
BG	Bulgaria	IE	Ireland	NZ	New Zealand
BJ	Benin	IT	Italy	PL	Poland
BR	Brazil	JP	Japan	PT	Portugal
BY	Belarus	KE	Kenya	RO	Romania
CA	Canada	KG	Kyrgystan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo			SE	Sweden
CH	Switzerland	KR	Republic of Korea	SI	Slovenia
CI	Côte d'Ivoire	KZ	Kazakhstan	SK	Slovakia
CM	Cameroon	LI	Liechtenstein	SN	Senegal
CN	China	LK	Sri Lanka	TD	Chad
CS	Czechoslovakia	LU	Luxembourg	TG	Togo
CZ	Czech Republic	LV	Latvia	TJ	Tajikistan
DE	Germany	MC	Monaco	TT	Trinidad and Tobago
DK	Denmark	MD	Republic of Moldova	UA	Ukraine
ES	Spain	MG	Madagascar	US	United States of America
FI	Finland	ML	Mali	UZ	Uzbekistan
FR	France	MN	Mongolia	VN	Viet Nam
GA	Gabon				

Description

MULTI-LINGUAL COMPUTER PROGRAMS

Technical Field

This invention relates to a method and system for creating multi-lingual computer programs by dynamically loading messages.

Background of the Invention

Localization is the process of altering a computer program so it is appropriate for the intended geographic area or user group. For example, a computer program running on a computer system in the United States would typically communicate with a user in English, while the same computer program running on a computer system in France would typically communicate with a user in French. The two different versions of the computer program would be essentially the same except for the natural language employed by the program's user interface. If two different users, one German and one French, desired to use the same computer system, then computer programs running on the system would have to have to be localized for German and French users.

In the past, computer systems accommodated multi-lingual communication by storing different natural language versions of computer programs. Thus, a French user would load a French version of a computer program and a German user would load a German version of a computer program. If a user required that the computer system's operating system communicate in a certain natural language, then the user would initialize (re-boot) the computer system to install the correct natural language version of the operating system. Storing and loading two versions of the same operating system on a computer system is an inefficient use of resources.

Currently, software developers spend a great deal of time localizing a computer program because each message that is output to a user must be translated into the appropriate language. When output messages are stored within the code of a computer program, a developer must have access to the entire program to translate messages. This access requirement is inconvenient because most programs consist of many modules that are linked together to make an executable program.

Software developers use resource files to store output messages rather than storing the messages directly in the program's code. This method eases translation because the messages to be translated are together in one file. To output a message, a program would retrieve the message from the appropriate resource file. Different resource files are used for each natural language. Developers create multiple natural language versions of the program by carrying out the following steps: 1) create a new resource file by translating the messages in an existing resource file into a desired natural language; 2) compile the new resource file; and 3) create an executable file by linking the compiled program modules with the translated, compiled resource file. Each natural language version of the program requires an executable file that contains all of the compiled program modules and the compiled resource file linked together.

Figure 1 is a block diagram of sample prior art executable files 101, 102, and 103. Executable file 101 represents a French version of the program Microsoft Excel. EXCEL.OBJ 104 represents all of the compiled program modules that make up Microsoft Excel. EXLFR.LIB 105 represents a compiled resource file containing all of the messages associated with Microsoft Excel. The executable file 101 consists of the compiled program modules 104 linked with the compiled resource file 105. Executable programs 102 and 103 represent English and German versions, respectively, of Microsoft Excel. Note how the same compiled program modules 104 are present in each of the executable programs 101, 102, and 103. Because the compiled program modules 104 are quite large (approximately 2 megabytes), it is wasteful to require that each natural language version contain the compiled program modules.

Summary of the Invention

The present invention provides a method and system for creating multi-lingual computer programs using dynamic message loading. In a preferred embodiment, a user of a computer system specifies a preferred language in which the user would like a computer program to communicate. The computer program communicates with the user by directing the computer system to output messages. The computer program has one or more associated message sets, which are lists of messages used by the computer program, each message set in a unique natural language. A message file containing message sets for one or more computer programs is stored on the computer system.

Alternatively, message sets associated with a computer program can be stored in a header area of the computer program. Preferably, a message set in a default natural language is stored in the header area of the computer program.

After the computer program is invoked, a Localizer searches the message file for a message set in the preferred language and associated with the computer program. Alternatively, before searching the message file, the Localizer searches the header area of the computer program for a message set in the preferred natural language. If a message set is not located in the header area, the Localizer then searches the message file. If the Localizer does not locate a message set in the preferred language in either the header area of the computer program or in the message file, the Localizer searches the header area of the computer program for a message set in the default language. After selecting the message set (either preferred or default), the Localizer makes a memory allocation request, requesting enough memory from the computer system to load the selected message set. The Localizer then loads the selected message set into the allocated memory and passes the address of the allocated memory to the computer program. After the message set in the preferred or default language is loaded into the computer system's memory, the Localizer returns control to the computer program.

Brief Description of the Drawings

Figure 1 is a block diagram of sample prior art executable programs.

Figure 2 is a block diagram of a computer system used in a preferred embodiment of the present invention.

Figure 3 is a detailed flow diagram of a method used in a preferred embodiment of the present invention.

Detailed Description of the Invention

The present invention includes a method and system for creating multi-lingual computer programs by dynamically loading messages. A computer program running on a computer system directs the computer to communicate with a user through the use of messages. Computer systems commonly communicate with a user by displaying output message to the user via a display device attached to the computer system and by receiving input messages from the user via a keyboard attached to the computer system. Other methods of receiving and outputting messages are known to those in the computer field.

In a preferred embodiment, the messages associated with a computer program are not stored within the code of the program. Instead, the messages are stored separately from the computer program so that the messages can be easily translated into other natural languages. Messages (in every supported natural language) are preferably contained in a message file, which is stored on the computer system.

The message file contains a plurality of message sets, with each set containing all messages associated with a computer program in a unique natural language. A message identifier is used within the code of the computer program to refer to a message stored within the message file. In an alternate embodiment of the present invention, message sets can be stored in a header area of the computer program rather than in the message file. The header area is a block of data containing details about the program and is usually found at the beginning of the program. The header area is not loaded into memory when the program is invoked.

Before a user invokes a computer program on the computer system, the user specifies a preferred natural language in which the user would like to communicate with the computer program. When the computer program is invoked, a Localizer provided by the present invention searches the message file, selecting messages that are associated with the invoked computer program in the preferred natural language. The Localizer then loads the selected messages into the computer system's memory. As the computer program executes, it references the selected messages in memory.

Because messages associated with a computer program are not stored within the program or linked to the compiled program, a developer may modify or translate messages without having access to the program. As long as the message identifier within an instruction of the program can be matched to a message identifier within a particular message set, the body of the executable program is independent of the messages.

Figure 2 is a diagram of a computer system 201 used in a preferred embodiment of the present invention. The computer system 201 contains a memory device 202 and a storage device 203. A plurality of executable programs can be stored on the storage device 203. Program.exe 204 is an example of such an executable program. Program.exe 204 has two parts, a program header 204a and a program body 204b. As explained above, the program header 204a is a block of data containing the size, location, and other details about Program.exe 204. The program body 204b contains instructions written in a binary format so that they can be loaded into the memory device 202 and executed by the computer system.

In a preferred embodiment, a message file containing a plurality of message sets is also stored on the storage device 203. Message.sys 205 is an example of such a message file. Message.sys 205 comprises a plurality of message sets, each message set containing the messages associated with a computer program. More than one message set can be associated with each computer program that is stored on the storage device 203. For example, message sets 1 through N are all associated with Program.exe 204. Each message set (1-N) contains the messages associated with Program.exe 204 in a unique natural language.

Each message set is made up of a message header and a message list. Similar to a program header, a message header is a block of data containing information about a message set. A message set is always preceded by a message header. Table 1 shows the contents of a message header used in a preferred embodiment of the present invention.

TABLE 1

MESSAGE_HEADER		
Comp_Size	Word	;Size of the compressed message list
Exp_Size	Word	;Size of the decompressed message list
Lang_Code	Byte(3)	;3 character Language Code
Country_Id	Word	;Country Identification number
Code_page	Word	;Code page number
Prog_Name	Word	;Name of program that uses messages
Signature	Word	;Signature(NS)
Reserved	byte(5)	;Reserved for future use.

Comp_Size indicates the size of the message list when the message list is compressed. Exp_Size indicates the size of the message list when the message list is not compressed. Lang_Code indicates in which natural language the message list is written. Country_Id indicates in which country the natural language identified in Lang_Code is spoken. Prog_Name indicates the name of the program or utility that uses the messages associated with this message header. Comp_Size, Exp_Size, Lang_Code, and Country_Id are all explained in more detail below. Code_page, Signature, and Reserved are not relevant to the present description and will not be discussed further.

Figure 3 is a detailed flow diagram of a method used in the preferred embodiment of the present invention to allow a user of a computer system to communicate with a computer program in a preferred natural language. When a user of a computer system invokes an executable computer program, a copy of the program's body is loaded into the computer system's memory device. In this example, the user has invoked Program.exe 204 so that a copy of Program.exe's body 204b has been loaded into the memory device 202. Also shown loaded into the memory device 202 is a copy of a Localizer 207, which is a computer program used to carry out the methods of the present invention (see Figure 2). Of course, the Localizer 207 could be made a part of another program (i.e., Program.exe) instead of being a separate program.

After the user has invoked Program.exe, in step 301 of Figure 3 the Localizer determines which natural language is the preferred natural language. The preferred natural language is the language in which the user of the computer system 201 prefers to communicate with Program.exe. Preferably, the user always specifies the preferred natural language before invoking a computer program, but it is not a necessity. In step 302 the Localizer searches the program header 204a for a message set in the preferred natural language. The Localizer only has to check the language code identifier in the message header of any message sets stored in program header 204a. If such a message set is not found in the program header 204b, then the Localizer searches the storage device 203 for the message file Message.sys 205. If Message.sys 205 cannot be found on the storage device 203, then the Localizer will prompt the user for the location of the message file. If the user cannot direct the Localizer to the location of the message file, then the Localizer searches the program header 204a for a message set in a default language. In a preferred embodiment, a message set in a default language is stored within a program header so that the invoked program has some means for communicating with the user. If the Localizer could not locate a message set in the preferred natural language or message set in a default natural language, then Program.exe would have no way of communicating with the user to tell the user that the message file 205 is not stored on the storage device 203.

After Message.sys 205 is located on the storage device 203, in step 305 the Localizer searches Message.sys file 205 for a message set in the preferred natural language. If such a message set is not found, the Localizer selects a message set in a default natural language from the program header 204a for the reasons stated above.

After steps 301 to 307 have been performed, the Localizer has located one of the following: a message set in the preferred natural language in the program header 204a; a message set in the preferred natural language in Message.sys 205; or a message set in a default natural language in the program header 204a. In step 308, the Localizer selects the located message set. In step 309, the Localizer makes a memory allocation request, requesting a block of memory from the memory device 202. In step 310, the Localizer loads the message list from the selected message set into the allocated memory. In step 311 the Localizer initializes Program.exe's data structures to point to the memory where the message list was loaded. The Localizer then returns control of the computer system to Program.exe. Program.exe communicates with the user of the computer system by referencing messages in the message list that is loaded into the computer system's memory device.

It can be seen then that the preferred embodiment described herein permits easy and efficient storage and selection of multiple natural language versions of a program. A first user of a computer system can specify the language in which the first user would like to communicate with a program, and then a second user can specify a different language in which the second user would like to communicate with the program.

Although the methods and systems of the present invention have been described in terms of a preferred embodiment, it is not intended that the invention be limited to this embodiment. Modification within the spirit of the invention will be apparent to those skilled in the art. The scope of the present invention is defined only by the claims that follow.

Claims

I claim:

1. A method of facilitating communication in a preferred natural language between a user of a computer system and a computer program running on the computer system, the computer system having a storage device and a memory device, the method comprising the steps of:

storing a message file on the storage device, the message file comprising a plurality of message sets, each message set comprising one or more messages that are associated with the computer program in a unique natural language;

determining the preferred natural language;

searching the message file to locate one message set containing messages associated with the computer program in the preferred natural language;

loading the located message set at a location on the memory device; and

notifying the computer program of where in the memory device the located message set is loaded.

2. The method of claim 1 including, prior to the step of determining the preferred natural language, the step of storing input from the user of the computer system on the storage device, the input comprising the preferred natural language.

3. The method of claim 1 wherein the computer program contains data structures for identifying a location on the memory device of a message set associated with the computer program and wherein the step of notifying the computer program includes initializing the computer program's data structures to point to the location on the memory device of the located message set.

4. The method of claim 1 wherein the plurality of messages sets in the message file are stored in a compressed manner and wherein the step of loading the located messages set includes the step of decompressing the located message set.

5. A method of localizing output messages used by an executable program in a computer system, the computer system having a memory device and a storage device, the method comprising the steps of:

storing a message file on the storage device, wherein the message file comprises a plurality of message sets, each message set comprising a plurality of output messages used by the executable program in a unique natural language;

invoking the executable program;

determining a current natural language;

retrieving a message set from the message file, the retrieved message set comprising output messages used by the executable program in the current natural language;

after retrieving the message set, requesting a block of memory from the memory device in which to load the retrieved message set;

loading the retrieved message set into the block of memory; and

notifying the executable program of where the retrieved message set is loaded.

6. The method of claim 5 wherein the retrieved message set is compressed and the step of loading the retrieved message set includes the step of decompressing the retrieved message set.

7. The method of claim 5 wherein each message set has a message header and a message list, each message header comprising control information such as a compressed size and a

decompressed size for the message list, and wherein the step of loading the retrieved message set includes the steps of:

examining the message header of the retrieved message set to compare the compressed size and the decompressed size for the message list; and

decompressing the message list when the compressed size is smaller than the decompressed size.

8. The method of claim 5 including the steps of prompting a user of the computer system to specify a current natural language and storing on the storage device the current natural language.

9. The method of claim 5 wherein the executable program comprises a header and a body, wherein the body is loaded into the memory device when the executable program is invoked on the computer system, including, subsequent to the step of determining a current natural language, the steps of:

examining the header to locate a message set in the current natural language;

when the header contains the message set in the current natural language, retrieving the message set from the header; and

when the header does not contain the message set in the current natural language, retrieving the message set from the message file.

10. The method of claim 5 wherein the step of retrieving a message set includes the additional steps of:

determining which output messages are used by the executable program; and

retrieving only the output messages used by the executable program from the message set.

11. The method of claim 5 wherein the message file has a location on the storage device, and wherein the method

includes, prior to the step of retrieving a message set, the step of locating the message file on the storage device.

12. A computer system for communicating with a user in a preferred natural language, the computer system having a computer program running thereon, storage device and a memory device, the computer system comprising:

means for storing a message file on the storage device, the message file comprising a plurality of message sets, each message set comprising one or more messages that are associated with the computer program in a unique natural language;

means for determining the preferred natural language;

means for searching the message file to locate one message set containing messages associated with the computer program in the preferred natural language;

means for loading the located message set at a location on the memory device; and

means for notifying the computer program of where in the memory device the located message set is loaded.

13. The computer system of claim 12 including means for storing input from the user of the computer system on the storage device, the input comprising the preferred natural language.

14. The computer system of claim 12 wherein the computer program contains data structures for identifying a location on the memory device of a message set associated with the computer program and wherein the means for notifying the computer program includes means for initializing the computer program's data structures to point to the location on the memory device of the located message set.

15. The computer system of claim 12 wherein the plurality of message sets in the message file are stored in a

compressed manner and wherein the means for loading the located messages set includes means for decompressing the located message set.

16. A computer system for localizing output messages used by an executable program in the computer system, the computer system comprising:

a memory device;

a storage device;

means for storing a message file on the storage device, wherein the message file comprises a plurality of message sets, each message set comprising output messages used by the executable program in a unique natural language;

means for invoking the executable program;

means for determining a current natural language;

means for retrieving a message set from the message file, the retrieved message set comprising output messages used by the executable program in the current natural language;

means for requesting a block of memory from the memory device in which to load the retrieved message set;

means for loading the retrieved message set into the block of memory; and

means for notifying the executable program of where the retrieved message set is loaded.

Figure 1
(Prior Art)

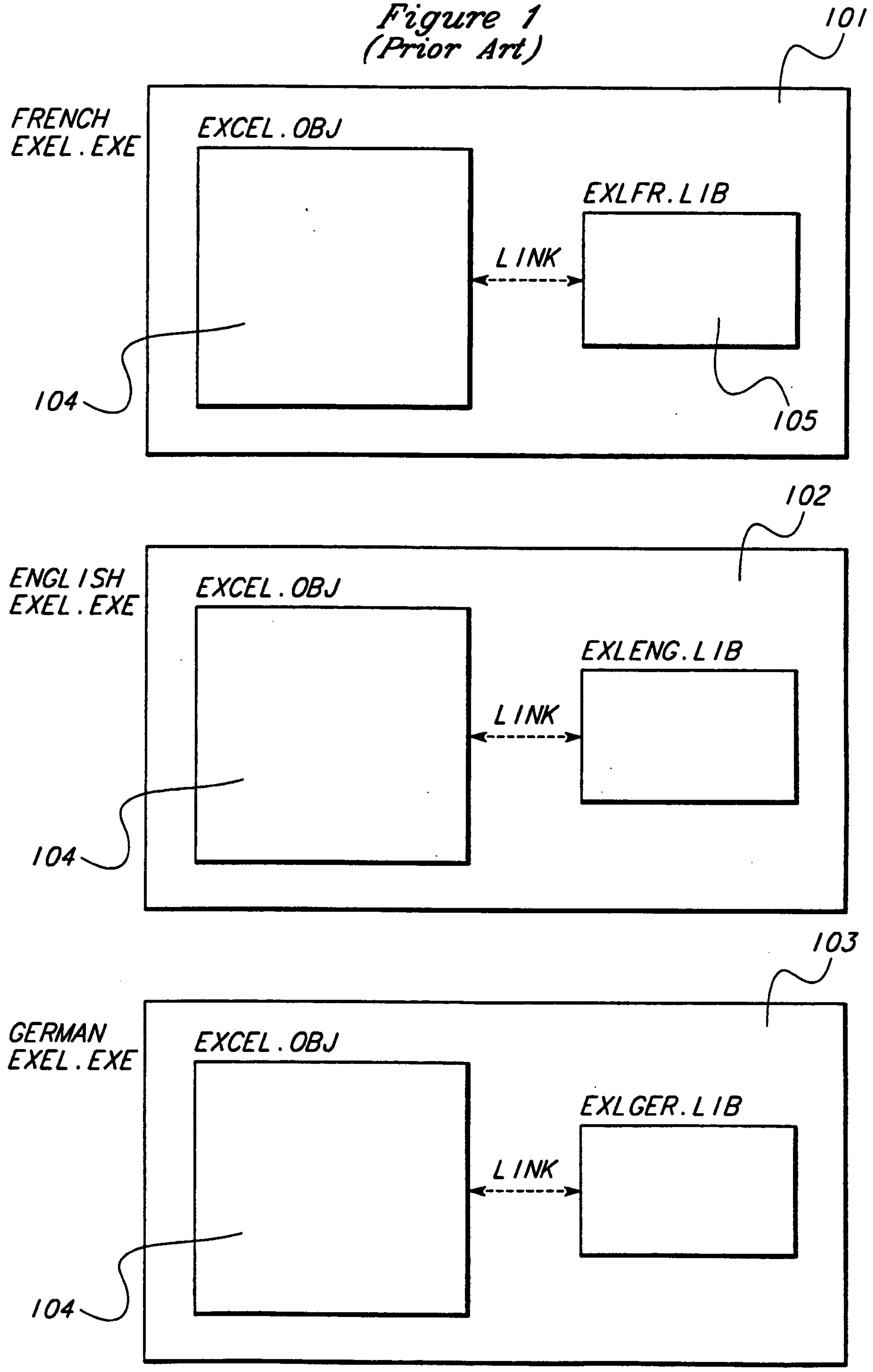
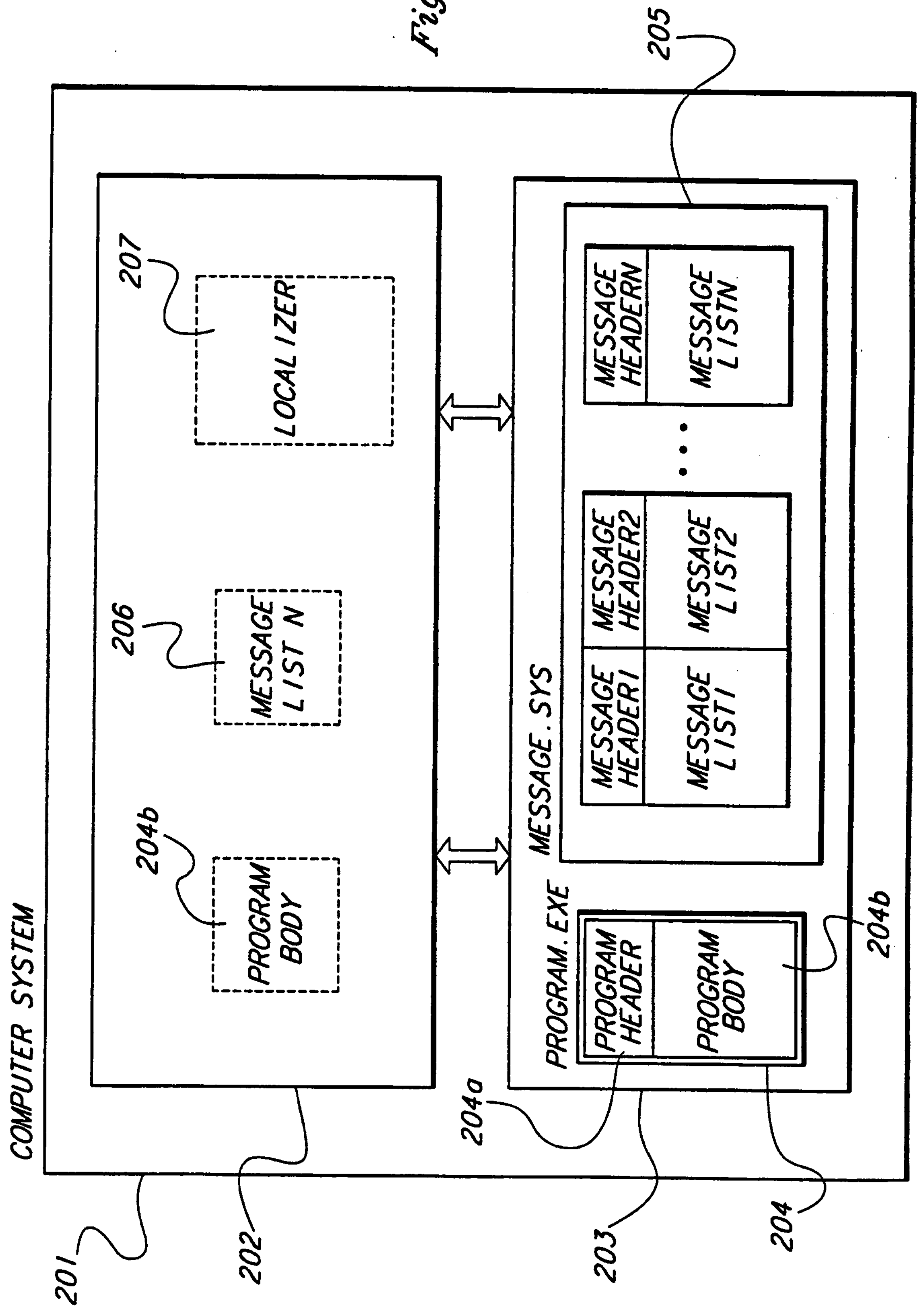


Figure 2


COMPUTER SYSTEM
PROGRAM BODY
MESSAGE LIST N
LOCALIZER
MESSAGE.SYS
MESSAGE HEADER1
MESSAGE LIST1
MESSAGE HEADER2
MESSAGE LIST2
MESSAGE HEADERN
MESSAGE LISTN
PROGRAM.EXE
PROGRAM HEADER
PROGRAM BODY
201
202
203
204
204a
204b
204b
205
206
207

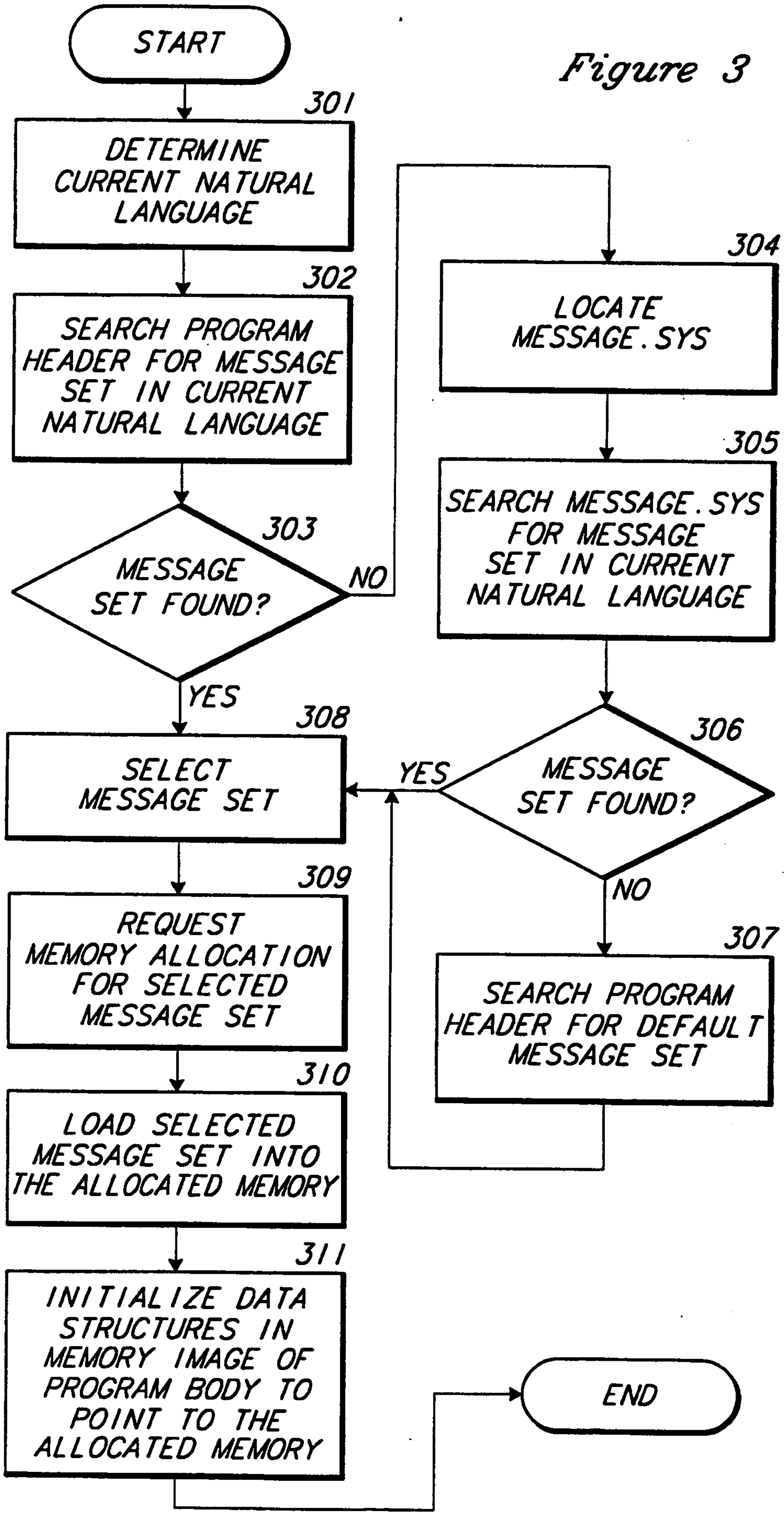

START
301
DETERMINE CURRENT NATURAL LANGUAGE
302
SEARCH PROGRAM HEADER FOR MESSAGE SET IN CURRENT NATURAL LANGUAGE
303
MESSAGE SET FOUND?
NO
YES
304
LOCATE MESSAGE.SYS
305
SEARCH MESSAGE.SYS FOR MESSAGE SET IN CURRENT NATURAL LANGUAGE
306
MESSAGE SET FOUND?
YES
NO
307
SEARCH PROGRAM HEADER FOR DEFAULT MESSAGE SET
308
SELECT MESSAGE SET
309
REQUEST MEMORY ALLOCATION FOR SELECTED MESSAGE SET
310
LOAD SELECTED MESSAGE SET INTO THE ALLOCATED MEMORY
311
INITIALIZE DATA STRUCTURES IN MEMORY IMAGE OF PROGRAM BODY TO POINT TO THE ALLOCATED MEMORY
END

Figure 3

INTERNATIONAL SEARCH REPORT

Intern al Application No
PCT/US 93/11061

A. CLASSIFICATION OF SUB ATTER
IPC 5 G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 5 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP,A,0 426 909 (HEIKKI MARTTILA OY) 15 May 1991 see column 1, line 31 - line 50 see column 2, line 33 - line 51 ---	1-16
A	EP,A,0 335 139 (IBM) 4 October 1989 see column 3, line 47 - column 4, line 15	1-16
A	see column 7, line 37 - line 43 ---	10
X	PATENT ABSTRACTS OF JAPAN vol. 10, no. 361 (P-523)4 December 1986 & JP,A,61 157 919 (CANON) 17 July 1986 see abstract -----	1-16

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

1

Date of the actual completion of the international search 2 March 1994	Date of mailing of the international search report 15.03.94
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 340-3016	Authorized officer Brandt, J

BEST AVAILABLE COPY

INTERNATIONAL SEARCH REPORT

..formation on patent family members

Intern al Application No
PCT/US 93/11061

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
EP-A-0426909	15-05-91	NONE		
EP-A-0335139	04-10-89	JP-A-	1270125	27-10-89